Fast Multi Class Distance Transforms for Video Surveillance

Theo E. Schouten[a], and Egon L. van den Broek[b]

[a]Institute for Computing and Information Science (ICIS), Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
T.Schouten@cs.ru.nl http://www.cs.ru.nl/T.Schouten

[b]Center for Telematics and Information Technology (CTIT), University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
vandenbroek@acm.org http://eidetic.ai.ru.nl/egon

ABSTRACT

A distance transformation (DT) takes a binary image as input and generates a distance map image in which the value of each pixel is its distance to a given set of object pixels in the binary image. In this research, DT's for multi class data (MCDTs) are developed which generate both a distance map and a class map containing for each pixel the class of the closest object. Results indicate that the MCDT based on the Fast Exact Euclidean Distance (FEED) method is a factor 2 tot 4 faster than MCDTs based on exact or semi-exact euclidean distance (ED) transformations, and is only a factor 2 to 4 slower than the MCDT based on the crude city-block approximation of the ED. In the second part of this research, the MCDTs were adapted such that they could be used for the fast generation of distance and class maps for video sequences. The frames of the sequences contain a number of fixed objects and a moving object, where each object has a separate label. Results show that the FEED based version is a factor 2 to 3.5 faster than the fastest of all the other video-MCDTs which is based on the chamfer 3,4 distance measure. FEED is even a factor 3.5 to 10 faster than another fast exact ED transformation. With video, multi class FEED it will be possible to measure distances from a moving object to various identified stationary objects with nearly the frame rate of a webcam. This will be very useful when the risk exists that objects move outside surveillance limits.

Keywords: distance maps, video surveillance, multi class data, classification, Fast Exact Euclidean Distance (FEED)

1. INTRODUCTION

A Distance Transformation (DT)[15] calculates an image (also called a distance map) in which the value of each pixel is its distance (according a given distance metric) to a given set of pixels O in the original binary image:

$$D(p) \;=\; min\{dist(p,q)\;,\; q \;\in\; O\} \tag{1}$$

The set pixels is called here O because often it consists of the pixels of objects, but it might as well consist of the background pixels or of data points in a certain feature space.

Distance maps can be applied in a range of settings, either by itself or as an important intermediate or auxiliary method in many applications; e.g., robot navigation[10], trajectory planning[25], skeletonization[11], Voronoi tessellations[8], fMRI data analysis[12], neuromorphometry[4], volume rendering, reconstruction of surface normals, and penetration of distances for applications in haptics and physics-based modeling[7], Bouligand-Minkowsky fractal dimension[3], Watershed algorithms[14], and video surveillance[19]. Note that for most applications, the Euclidean Distance Transformation (EDT) will produce the most accurate results.

Rosenfeld and Pfaltz[15,16] introduced in 1966 the first fast DT for the city-block and chessboard distance measures. These distance maps are build up during two raster scans over the image using only local information; i.e., distances to neighboring pixels. Ten years later, Borgefors[2] extended them to a number of chamfer DTs, where during the scans different weights are given to neighboring pixels to produce better approximations of the Euclidean Distance. In 1998 Shih and Liu[21] presented their method to obtain semi-exact EDTs. They started with four scans on the image. Next, a look-up table method was used to correct the wrong pixels. For a large

majority of cases, they were able to determine exact EDTs. In 2004 Shih and Wu[22] introduced a similar method which uses only two scan over the image and which is therefore faster.

In 2003 Maurer, Qi and Raghavan[13] obtained an exact EDT, based on dimensionality reduction and partial Voronoi diagram construction. Their method is applicable to binary images of any dimension. A year later, Schouten and Van den Broek[20] presented their Fast Exact Euclidean Distance (FEED) transformation. With FEED we introduced an algorithm, which obtained an exact EDT in a computational very cheap way. FEED was applied to 2 and 3 dimensional images[18,20], it is generally faster than the method of Maurer, Qi and Raghavan[13], but the code is longer and more complex.

In this paper we describe the extension of several DTs to handle the case that the set of objects in the input image is labeled. Such a Multi Class Distance Transformation (MCDT) produces, besides a distance map, also a class map in which each pixel obtains the label of the closest object. A preliminary such extension for FEED was presented previously[23,24].

In 2005 Schouten, Kuppens and Van den Broek[17] introduced an adaption of FEED and several other DTs in other to generate distance maps for a video sequence with moving and fixed objects in a faster way. This was used later by the same authors[19] to detect motion; i.e. extract information about the relative positions of stationary and moving objects nearly real-time in video sequences. The resulting distances between the objects can, for example, be used for surveillance purposes. Unwanted or possible dangerous distances (e.g., too small or too large) can be automatically detected, in order to take appropriate actions. These actions might include intervention by a (human) supervisor watching the video or sending relevant distance information to objects (or persons) that have capabilities for changing the course of their actions[1,6,9]. The automation of this process is of the utmost importance since the vigilance of humans is limited.

In this paper we also present the adaption of the developed MCDTs such that they can be used to faster generate distance and class maps for video images. Surveillance limits can then be set on identified objects increasing the usability of such video surveillance systems. It will be shown that of the developed video-MCDTs the FEED version is by far the fastest, faster even than video-MCDTs based on crude approximations of the ED.

In Section 2 we briefly discuss the principles of the original DTs used in this paper and their extension to MCDTs is discussed in Section 3. The test environment, consisting of 3 computer systems and set of images of different sizes, is given in Section 4, as well as the results obtained on them by the MCDTs. Following in Section 5is the description of the adaption of the developed MCDTs to faster generate distance and class maps for video frames costing of several fixed objects and one moving object. Test results on two video sequences are described in Section 6. Finally in Section 7 the work presented in this paper is discussed.

2. USED ORIGINAL DISTANCE TRANSFORMATIONS

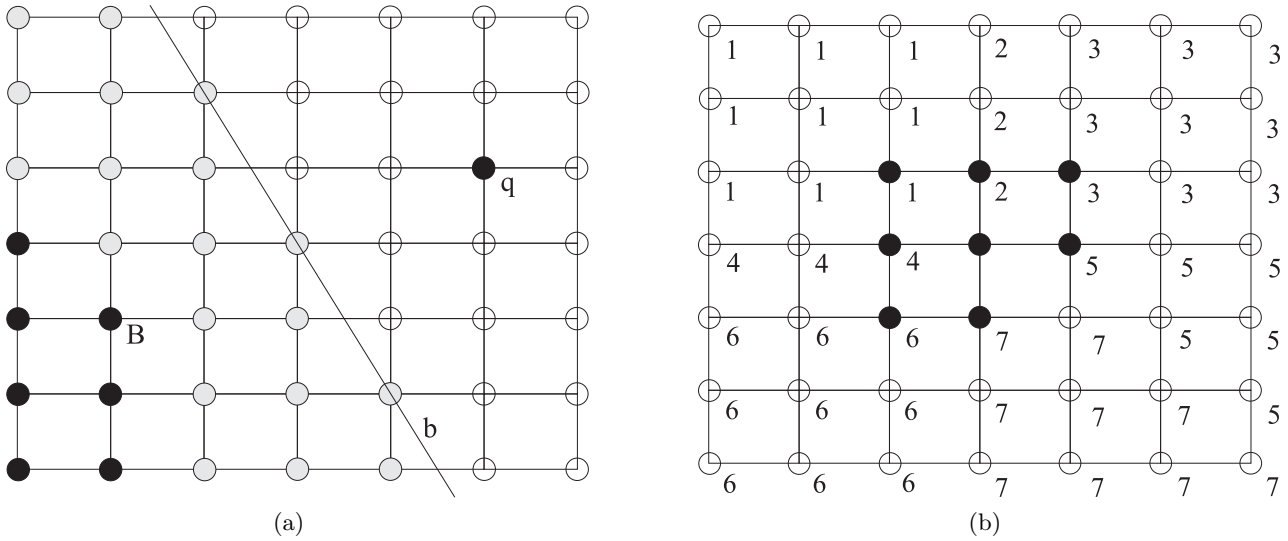
In this section the original DTs are discussed that in the rest of this paper are further developed into MCDTs and video-MCDTs. Special attention is paid to characteristics, like memory usage, that are important for execution times and their scaling with image size on different execution platforms.

The city-block DT of Rosenfeld and Pfaltz[15,16] and the chamfer 3,4 DT of Borgefors[2] are included here for comparison reasons. They are denoted in the rest of this paper as CH11 (the city-block is also a chamfer method) resp. CH34. Although they provide less accurate results, they are fast and simple and that might be of interest for certain applications. They obtain their results in two raster scans over the image and don't use auxiliary matrices. The fastest execution is obtained by doing all the calculations in integer, obtaining integer distances for CH11 and integer distances multiplied by three for CH34.

Both the four scan method of Shih and Liu[21] and the later two scan method of Shih and Wu[22] are included and denoted as EDT4 resp. EDT2. The scans alone produce an approximated ED in the sense that for most pixels the obtained distance is correct but sometimes (in a few percent of the pixels) it is a bit to high. This is due to the fact[5,17] that the tiles of the Voronoi diagram are not always connected sets on a discrete lattice. In previous experiments[17,20] it was shown that EDT2 is faster but less accurate than EDT4. As EDT4 is slower than FEED, the correction method of Shih and Liu[21] to obtain more accurate results was not implemented. In

contrast to the chamfer methods, these methods need two auxiliary matrices with size equal to the image size. These matrices contain the absolute x and y distances to the current closest object point and therefore the size of each element of them must be large enough to contain the largest coordinate in the image. Like the input and output image, they are processed in a raster scans during the execution of the algorithm. Calculations are performed in integer, obtaining as final results the square of the (nearly) EDs.

Further the method of Maurer, Qi and Raghavan[13] is used here, denoted as EDLT. The last two letters come of the name from Linear Time as the authors describe their algorithm as having linear time in the total number of pixels. They prove that the number of integer arithmetic operations is indeed proportional to the number of pixels, but with modern computer systems this might, over a given range of image sizes, not translate into linear time. This depends on the efficiency of with the various caches in a computer system are used. Also in this algorithm the calculations are performed in integer, obtaining as final results the square of the exact EDs. The algorithm, after an initialization scan, first performs a row-wise and then a column-wise raster scan over the image. Note that the latter might introduce cache inefficiencies. For each row or column a partial Voronoi diagram is constructed and used to calculate intermediate and final squared EDs from. Auxiliary matrices are not used, only 2 auxiliary vectors, with size equal to the maximum of the sizes in the x and y dimension of the image, are used for the partial Voronoi diagram. All the given (see Section 5 of their paper) efficiency remarks are implemented and a further speedup was obtained by integrating the computation of D_0 and D_1. The resulting code is simple and small, around 60 lines of code; for comparison CH11 has about 80 and EDT4 about 180 lines of code.



(a) (b)

Figure 1: Principle of limiting the number of background pixels to update. (a) Only pixels on and to the left of the bisection line b between a border pixel B and an object pixel q have to be updated. (b) An example showing that each background pixel has to be updated only once. Each background pixel is labeled by the border pixel which updates it.

In the FEED method of Schouten and Van den Broek[17,19,20] each object pixel *feeds* its ED to all pixels in the image, which in turn calculate the minimum of all received EDs. In its naive form:

$$
\begin{array}{ll}
(1) & \mathit{initialize}\ D(p)\ =\ \mathit{if}\ (p\ \in\ O)\ \mathit{then}\ 0,\ \mathit{else}\ \infty \\
(2) & \mathit{for\ each}\ q \in O \\
(3) & \quad \mathit{for\ each}\ p \\
(4) & \quad\quad \mathit{update}:\ D(p)\ =\ \mathit{min}\{D(p)\ ,\ ED(q,p)\}
\end{array}
\tag{2}
$$

Speedup of this naive algorithm is achieved in various ways. In line (2) only the "border" pixels B of O have to be considered because the minimal ED from any background pixel to the set O, is the distance from that background pixel to a border pixel B of O. A border pixel B is defined here as an object pixel with at least one

of its four 4-connected pixels in the background. Moreover, the number of pixels that have to be considered in line (3) can be limited to only those that have an equal or smaller distance to the current B than to any object pixel q. In principle, by taking all q into account, it could be assured that each background pixel is updated only once. See Figure 1 for a graphical explanation of it. But then the bookkeeping of which background pixels to update would take much longer than the time gained by not performing unneeded updates. The complexity of the FEED lies in using only those q that are easy to find, easy to bookkeep and that have the largest effect on removing unneeded updates. The fastest implementation has about 450 lines of code, uses many parameters for adjustment and three auxiliary vectors, with size equal to the maximum size of the image. Further the ED is taken from a pre-computed matrix with a size equal to the image size. The contents of this matrix can be changed to any non-decreasing function of the ED, like the square of the ED or truncated or ceiled integer ED values. Direct calculation of the square of the ED might be faster than using a matrix, as was the case with a 3-D implementation[18] of FEED.

3. MULTI CLASS DISTANCE TRANSFORMATIONS

In this section the extension of the DTs described in the previous section to MCDTs is described. Each MCDT now has an additional input image, containing for each object pixel the class of the object from 1 to the maximum number of classes and for each background pixel the value 0. In the additional output image each pixel receives the class of the nearest object according the distance measure used in the DT.

For the CH11 and CH34 DTs the distance for each pixel during each raster scan is derived from the distances of neighboring pixels that are already visited in that scan. The class of each pixel can therefore simply be set to the class of the neighboring pixel from which the distance is derived. This gives for example for CH11 code fragments like:

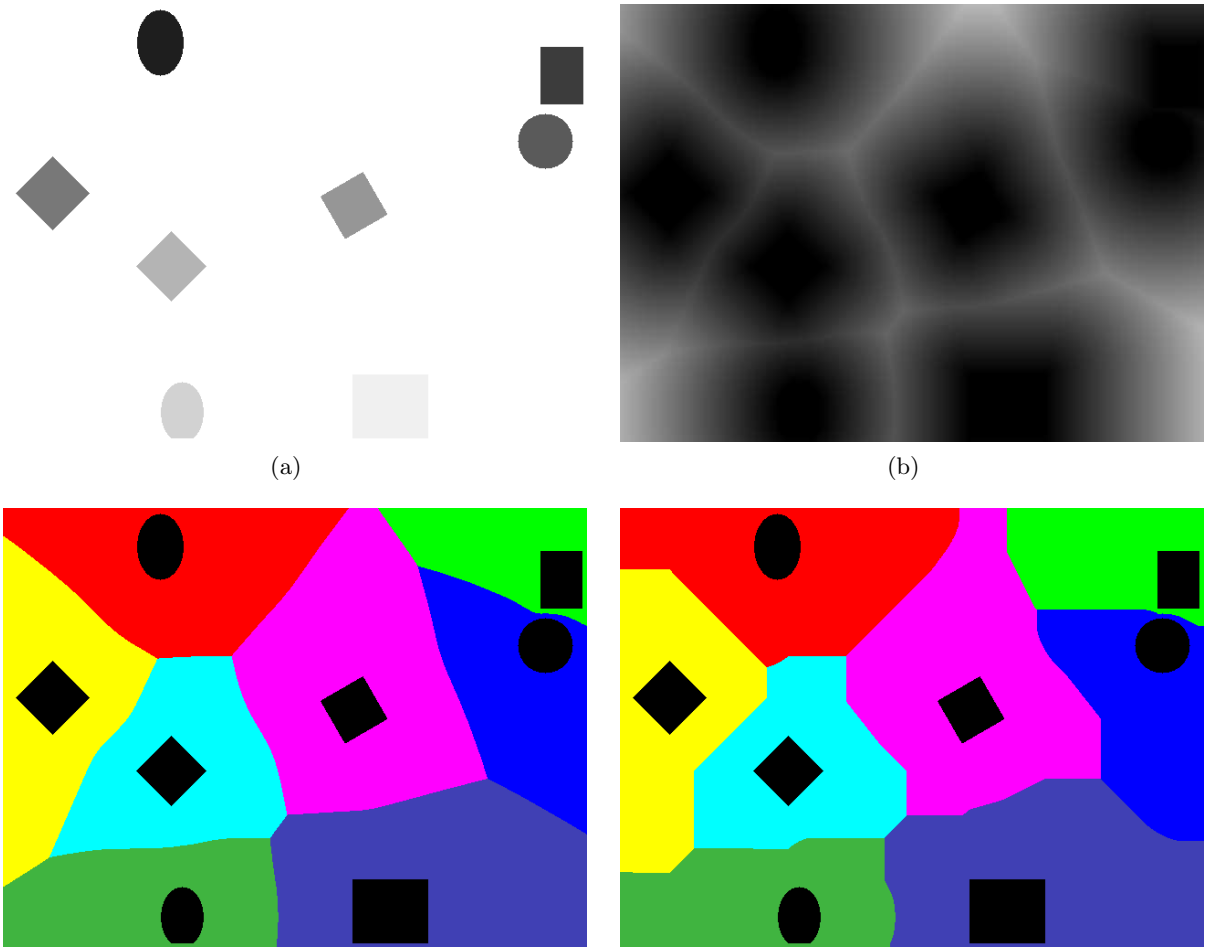
```
#define D5 1+ed[y  ][x+1]
#define D7 1+ed[y+1][x  ]
  if( ed[y][x] == 0) continue;  // an object pixel has distance 0
  if(D5 < D7) {
    if( D5 < ed[y][x]) {ed[y][x]=D5;oc[y][x]=oc[y][x+1];}
  } else {
    if( D7 < ed[y][x]) {ed[y][x]=D7;oc[y][x]=oc[y+1][x];}
  }
```

in the backward scan from the highest to the lowest y and x coordinates. Here *ed* is the output image receiving the distance and *oc* is receiving the class. This means only one extra assignment per pixel, an increase in execution time can be more expected from stepping through an additional matrix certainly when this invokes additional cache fills.

For EDT4 and EDT2 the same method as for the chamfer DTs can be applied. The derivation of the distance for each pixel during the scans involves much more arithmetic operations than for the chamfer methods; therefore the increase in execution time is expected to be smaller than for the chamfer methods.

For EDLT during initialization also the class of the object pixels in the output class image has to be initialized in line 4 of Figure 4 of the paper[13]. For the partial Voronoi diagram an extra vector is used that is set to the class of the Voronoi points in lines 6 and 11 of Figure 5. In line 23 of Figure 5 the output class pixel is set to the class of the same Voronoi point as used for setting the distance output. All these additions require only a few additional assignments per pixel, but more memory locations are addressed which may lead to less efficient uses of the cache systems of actual computer systems.

For FEED, besides initializing the class of object pixels in the class map, the update step in Equation 2 has to be changed: if $ED(q,p)$ is smaller than $D(p)$ also the class of p is set to the class of q. Also here a few additional assignments per pixel, equal to the average number of updates per pixel. The stepping through an additional matrix might induce a larger increase in execution time.



(a) (b)

(c) (d)

Figure 2: (a) An input image of size 640x480, the intensity indicates the class of the objects. (b)The resulting ED map as determined by the FEED and EDLT methods. (c)The resulting colored class map according the ED as given by both the MC-FEED and MC-EDLT methods. (d)The resulting colored class map according the city-block distance determined by the MC-CH11 method.

4. TEST ENVIRONMENTS AND RESULTS

For testing the DT and MCDT methods described in the previous sections, three different execution platforms have been used:

- AMD: an AMD Athlon XP® 2100+ CPU running at 1735 MHz with 64KB L1 I-cache, 64KB L1 D-Cache, 256 KB L2 Cache and 512 MB memory.
- P-4: an Intel Pentium® 4 3.00 GHz CPU running at 2990 MHz with 12 KuOps L1 T-Cache, 16 KB L1 D-Cache, 2048 KB L2 Cache and 1024 MB memory.
- P-M: an Intel Pentium® M 1.60 GHz CPU running at 1598 MHz with 32 KB L1 I-cache, 32 KB L1 D-Cache, 2048 KB L2 Cache and 512 MB memory.

On all systems the Microsoft® Visual C++ 6.0 programming environment in the standard release setting was used.

	timing on AMD			timing on P-4			timing on P-M			Average error		
DT	I-s	I-m	I-l	I-s	I-m	I-l	I-s	I-m	I-l	I-s	I-m	I-l
FEED	0.051	0.069	0.084	0.026	0.030	0.037	0.025	0.036	0.048	0.0	0.0	0.0
EDLT	0.078	0.294	0.308	0.041	0.048	0.155	0.047	0.061	0.256	0.0	0.0	0.0
EDT2	0.100	0.189	0.192	0.052	0.052	0.053	0.090	0.094	0.101	0.080	0.020	0.536
EDT4	0.206	0.293	0.295	0.087	0.090	0.091	0.183	0.195	0.200	0.021	0.056	0.124
CH11	0.031	0.033	0.034	0.010	0.010	0.011	0.015	0.018	0.020	14.25	26.42	51.78
CH34	0.037	0.039	0.040	0.015	0.014	0.014	0.022	0.023	0.025	2.010	3.975	8.107

Table 1: Timing (in μs per pixel) and accuracy results for the DT methods on three different machines and for sets of 640x480 (I-s), 1280x960 (I-m) and 2560x1920 (I-l) images. The average error per pixel is relative to the euclidean distance.

	timing on AMD			timing on P-4			timing on P-M			% wrong pixels		
MCDT	I-s	I-m	I-l	I-s	I-m	I-l	I-s	I-m	I-l	I-s	I-m	I-l
FEED	0.064	0.084	0.195	0.033	0.037	0.045	0.036	0.048	0.119	0.0	0.0	0.0
EDLT	0.129	0.340	0.448	0.066	0.076	0.195	0.101	0.116	0.317	0.0	0.0	0.0
EDT2	0.110	0.199	0.202	0.059	0.060	0.066	0.100	0.104	0.111	0.255	0.187	0.266
EDT4	0.225	0.343	0.345	0.100	0.102	0.107	0.200	0.210	0.219	0.100	0.123	0.084
CH11	0.036	0.040	0.042	0.013	0.013	0.014	0.018	0.023	0.025	6.656	7.560	5.849
CH34	0.050	0.051	0.056	0.020	0.020	0.021	0.032	0.034	0.037	1.529	1.810	1.598

Table 2: Timing (in μs per pixel) and accuracy results for the MCDT methods on three different machines and for sets of 640x480 (I-s), 1280x960 (I-m) and 2560x1920 (I-l) images. The % wrong pixels is the percentage of pixels that receive a different class compared to the euclidean distance classification.

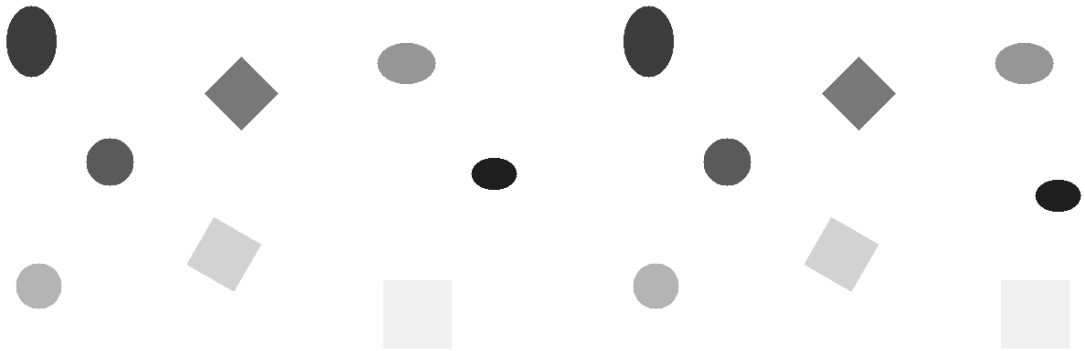
Various large sets of test images were generated. In this section timing and accuracy results on three of them are presented, other sets were used to experimentally verify the correctness of the FEED and EDLT implementations and their extensions to multi class and video versions. The first set, denoted as I-s, consists of images with size 640x480, a size often used in cameras for video surveillance. Eight objects were randomly placed on each image ensuring that they don't overlap or touch. Each object was randomly selected from a set of 8 types (square, rotated squares, circle, ellipses) with randomly selected diameter between 50 and 100 pixels. Each object received a different intensity value denoting the class of the object. The other two sets were scaled up versions to size 1280x960 (set I-m) and 2560x1920 (set I-l) images.

In Figure 2(a) an example of a 640x480 image is shown and Figure 2(c) gives its ED map, as determined by both the FEED and EDLT methods. Figure 2(c) shows the resulting colored class map as determined by both the MC-FEED and MC-EDLT methods. As contrast the class map obtained by the MC-CH11 method is given in Figure 2(d).

In Table 1 timing and accuracy results of the original DT methods are given, while Table 2 shows the results of the multi class versions. For this comparisons the output distance was computed by all methods in floating point, some of the methods could therefore be up to 10% slower than when using their natural output format. FEED is clearly faster than EDLT, the other exact ED, by 31% to 84% for the non-class versions and 50% to 77% for the class versions. Further FEED is faster than the semi-exact EDs (EDT2 and EDT4); and is only a factor 2 to 4 slower than the crude CH11 approximation of the ED. This is true for both the non-class and the class versions. Note further that the cache systems of the various machines have a large influence on the execution time, when increasing the size of the images a rather sudden increase in the time per pixel can occur. For actual images not only the number of arithmetic CPU operations is important but also how much one and two dimensional matrices are used concurrently and how steps through them are taken.

5. VIDEO MULTI CLASS DISTANCE TRANSFORMATIONS

In this section it is described how the MCDTs, developed in Section 3 are extended to faster produce their distance and class maps in case of a video sequence with fixed objects and one moving object. An example of



(a) (b)

Figure 3: Two following frames in a video sequence. (a) The first frame of size 640x480, the intensity indicates the class of the objects. (b) The next frame showing that the darkest object is the moving object.

two neighboring frames of such a sequence is shown in Figure 3. A first version of the extension for the DTs was already described previously[17], while also a system was described[19] that locates stationary and moving objects from a simulated actual video stream.

The distance and class maps for the fixed objects (D_{fixed} and C_{fixed}) and for the moving object (D_{moving} and C_{moving}) can be calculated separately and then be combined to obtain the distance and class maps for the total frame:

$$\begin{aligned} &D_{fixed+moving}(p) \ = \\ &\quad if(\ D_{fixed}(p) \ < \ D_{moving}(p)\ ) \\ &\quad then\ \{\ D_{fixed+moving}(p) \ = \ D_{fixed}(p)\ ;\ C_{fixed+moving}(p) \ = \ C_{fixed}(p)\ ;\} \\ &\quad else\ \{\ D_{fixed+moving}(p) \ = \ D_{moving}(p)\ ;\ C_{fixed+moving}(p) \ = \ C_{moving}(p)\ ;\} \end{aligned} \tag{3}$$

For each of the MCDTs the following procedure is used. For the first frame of a sequence the pixels of the moving object are set to be background pixels and then D_{fixed} and C_{fixed} are calculated. The maximum distance d_{max} of D_{fixed} is calculated because that also defines a maximum distance over which a pixel of a moving object can have an influence on D or C maps. Further the background pixels of the fixed images are encoded in run-length lists, one for the x and one for the y direction. These lists are used to speed up locating the moving object.

Then for each frame the moving object is located, using a refining sequence of scans over the image. The minimum and maximum x and y coordinates of the moving object are then determined and extended on all size by d_{max}. This defines the bounding box in which the MCDT is applied to determine D_{moving} and C_{moving}. After that the combining operation given in Equation 3 is applied only inside the bounding box to obtain the final distance and class maps for the frame. Some speed-up is achieved by using the same memory locations for D_{fixed} and $D_{fixed+moving}$ and then undoing the combining operation for the next frame using a simple memory copy from a saved version of D_{fixed}. The same is done for the C maps.

For FEED however a speed-up can be achieved by merging the application of FEED on the bounding box with the combining operation of Equation 3. This is not possible for the other methods because they produce during their first scans over the image only partial evaluations of their distances. In the partial FEED the output distance map is now initialized with the fixed map instead of the original initialization with 0 and ∞ distances (the class map is initialized with the class of the object pixels) and then the moving object is located. For each located border pixel of the moving object the search procedure of FEED is started for object pixels q that limit the area over which the border pixel has to $feed$ its ED, see the description of FEED in Section 2. The run-length lists, determined for the fixed objects, are now used to speed-up the original FEED search procedure. Further

d_{max} is used to limit the $feed$ area to a circle, approximated by its outside octagon. As the search for object pixels q is faster now, several parameters were adjusted to provide the optimum trade-off between searching and updating in order to achieve the minimum execution time.

6. TEST RESULTS FOR VIDEO MCDTS

	timing on AMD			timing on P-4			timing on P-M			Errors	
DT	full	fixed	video	full	fixed	video	full	fixed	video	average	class
FEED	0.071	0.077	0.008	0.035	0.039	0.003	0.041	0.045	0.004	0.0	0.0
EDLT	0.127	0.133	0.037	0.066	0.070	0.025	0.102	0.105	0.037	0.0	0.0
EDT2	0.111	0.116	0.043	0.060	0.064	0.026	0.102	0.106	0.040	0.108	0.132
EDT4	0.214	0.218	0.069	0.100	0.105	0.036	0.168	0.174	0.056	0.029	0.089
CH11	0.035	0.042	0.031	0.012	0.017	0.013	0.018	0.023	0.019	12.423	4.217
CH34	0.048	0.054	0.021	0.020	0.024	0.009	0.031	0.036	0.013	1.599	1.380

Table 3: Timing (in μs per pixel) and accuracy results for the video MCDT methods on three different machines for 640x480 frames with 7 fixed objects and 1 moving object. See Figure 3 for examples of the frames. The "full" time is the time to process a full fame with the original MCDT. The time needed for processing of the fixed objects in the first frame is given in the "fixed" column, while the "video" time gives the time needed to obtain distance and class results for each frame. The average distance error per pixel is relative to the euclidean distance. The class error is the percentage of pixels that receive a different class compared to the euclidean distance classification.

	timing on AMD			timing on P-4			timing on P-M			Errors	
DT	full	fixed	video	full	fixed	video	full	fixed	video	average	class
FEED	0.068	0.074	0.005	0.033	0.037	0.002	0.038	0.043	0.002	0.0	0.0
EDLT	0.110	0.111	0.019	0.058	0.062	0.012	0.091	0.095	0.019	0.0	0.0
EDT2	0.113	0.118	0.022	0.061	0.066	0.013	0.104	0.109	0.020	0.082	0.169
EDT4	0.217	0.223	0.035	0.103	0.107	0.018	0.174	0.178	0.028	0.032	0.095
CH11	0.036	0.043	0.016	0.012	0.017	0.007	0.018	0.023	0.009	11.298	2.520
CH34	0.049	0.055	0.011	0.020	0.025	0.004	0.031	0.036	0.006	1.315	1.028

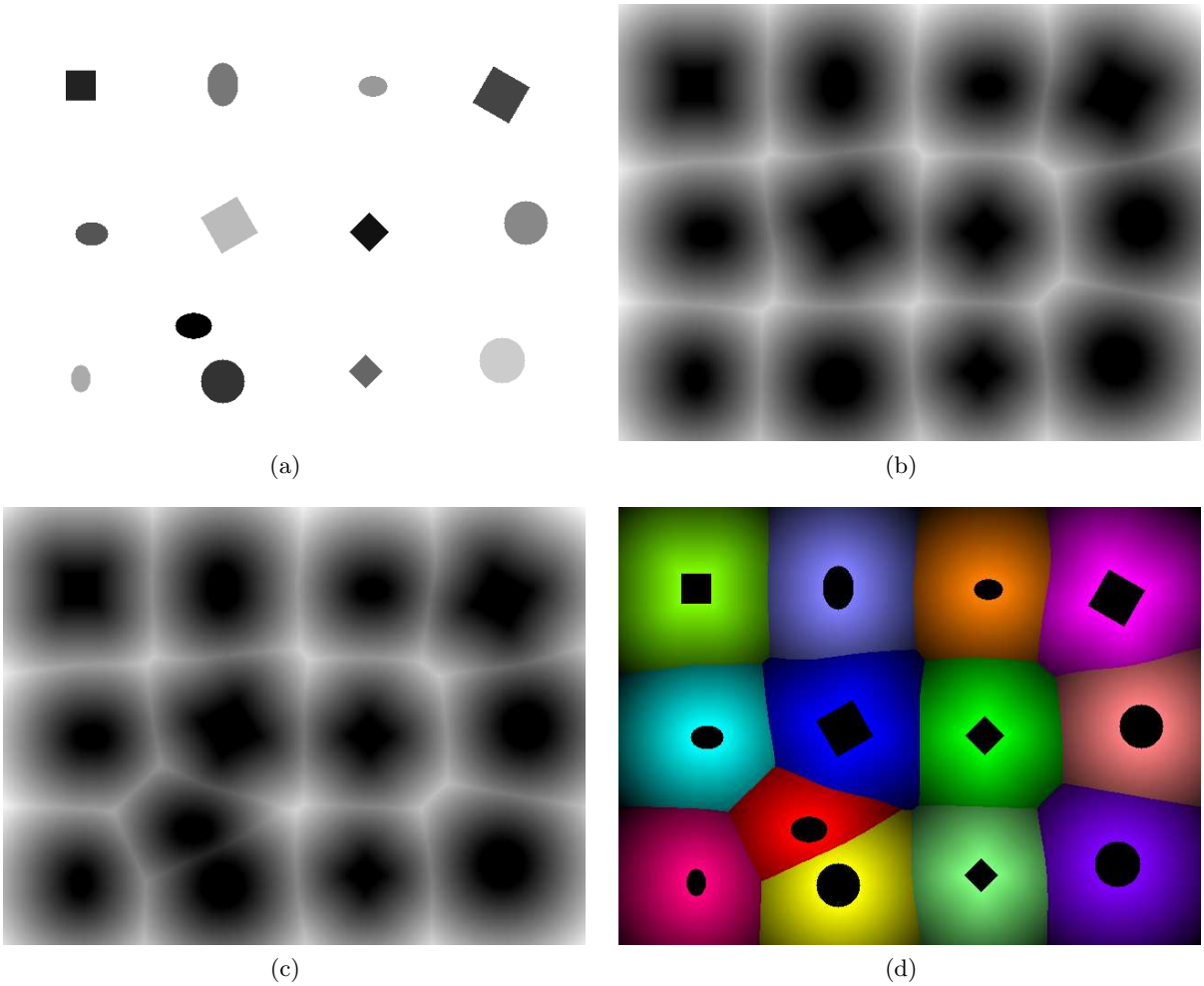
Table 4: Timing (in μs per pixel) and accuracy results as in Table 3 but for 640x480 frames with 12 fixed objects and 1 moving object.

In Table 3 timing (in μs per pixel) and accuracy results are given for the video MCDT methods developed in the previous section. The video sequence contained 640x480 frames with 7 fixed objects and 1 moving object, see Figure 3 for examples of the frames. Timing results (in μs per pixel) are given for the three different machines mentioned in Section 4. The "full" time is the time to process a full fame with the original MCDT. The time needed for processing of the fixed objects in the first frame, as described in the previous section, is given in the "fixed" column, while the "video" time gives the time needed to obtain the final distance and class results for each frame. The average distance error per pixel is relative to the euclidean distance and the class error is the percentage of pixels that receive a different class compared to the euclidean distance classification.

The results show that FEED is by far the fastest video MCDT. It is a factor 2.5 to 3.5, depending on the machine, faster in processing each frame than the fastest other video MCDT: the CH3,4 method, which also produces less accurate distance and class results. The fastest MCDT method on full images, the CH11 method, is slower than the CH3,4 method in processing frames, because it largely overestimates distances toward the 45° directions. This makes the needed bounding box around the moving object larger for the CH11 distance than for the CH34 distance. On 2 of the machines the video frame processing of CH11 is even slower than processing each frame fully. FEED is also a factor 4 to 10, depending on the machine, faster in processing each frame than MC-EDLT, the other exact ED method. Also here the effect of the cache systems on the execution time is large.

Further FEED gains more in execution time when going from full image to frame processing than all the other methods.

The time needed for processing the fixed objects, in order to start the fast processing of the frames, is only the time needed for a full image incremented by a very small amount of time for the run-length encoding of the fixed objects.



(a) (b) (c) (d)

Figure 4: (a) An input frame of size 640x480, the darkest ellipse is the moving object. (b)The resulting ED map of the fixed objects as determined by the FEED and EDLT methods. (c)The ED map of the frame, determined by the video, multi class FEED and EDLT methods. (d)The combined class and ED map. Each class has a different color. The darkness of each pixel is proportional to its ED.

In Table 4 results are given for 640x480 frames with 12 fixed objects and 1 moving object. Because the d_{max}, see the previous chapter, and hence the bounding box around the moving object is smaller, the times for processing the frames are smaller than for the frames with less objects. This gain in time is rather constant for all machines, for FEED it varies from a factor 1.57 to 1.60, but for the other methods from a factor 1.91 to 2.13. Again this confirms our earlier observation[17,20] that the speed of FEED is more dependent on the content of the images than the other methods. For these frames with 13 objects FEED is a factor 2.0 to 2.6 faster in processing each frame than the fastest other video MCDT: the CH3,4 method. FEED is now a factor 3.4 to 8, depending on the machine, faster in processing each frame than MC-EDLT, the other exact ED method.

In Figure 4 some results are shown for a 640x480 frame with 12 fixed and 1 moving object. The combined class and distance map shows a way to represent results of the developed methods useful for visual inspection.

7. DISCUSSION

In this paper we have extended a number of distance transformation methods to handle also also multi class data. Besides providing as output a distance map, these MCDT methods also provide a class map that gives for each pixel the class of the closest object.

The MCDT based on the Fast Exact Euclidean Distance (FEED) transformation developed previously by the authors[20], was shown to be fastest of all MCDTs based on exact or semi-exact (only a small percentage of the pixels receive a slightly wrong ED) ED transformations. It is also only a small factor slower than MCDT methods based on city-block and chamfer 3,4 approximations of the ED. This makes multi class FEED a good choice for developing applications in the area mentioned in the fist section of this paper.

Further we have extended and adapted the developed MCDT methods to increase the speed of obtaining distance and class maps of video sequences, where frames contain several fixed objects and one moving object. For the first frame distance and class maps are calculated for the stationary objects only and some information is derived in order to further speed-up the processing of each frame. This preprocessing takes only a bit more time than fully processing a frame. Then for each frame the maps are only calculated for the influence area of the moving object, these maps are then combined with the maps for the fixed objects in order to obtain the final results.

The video-MCDT based on FEED is shown to be faster than all the other ones, including even the city-block and chamfer 3,4 approximations of the ED. It is in our tests a factor 3 to 10 faster than the video-MCDT based on EDLT, the exact ED transformation method of Maurer, Qi and Raghavan[13]. This makes video, multi class FEED an excellent basis for developing video surveillance applications.

The large variation in the speed gain factor is due to the way the various cache systems in modern machine are exploited by the programs. This depends on the number of matrices and vectors uses, their sizes and the way the program steps trough them. This might have a larger effect than the number of arithmetic operations per pixel, up to now the most important characteristic to minimize when developing algorithms. It might be concluded that FEED handles the memory better than the EDLT method.

Although the principles of EDLT are difficult to understand, the resulting program is very small, a factor 7 less lines of code than FEED, which also contains a number of parameters to vary and optimize strategies to obtain the lowest execution time. This might be an advantage of EDLT when going to higher dimensional images for which both EDLT and FEED are suited. Also when moving to parallel or muti-core machines as execution platforms; both EDLT and FEED can easily be parallelized. But on the other hand, the search strategies used by FEED to limit the area over which EDs have to be "feed", are flexible and can be adapted to exploit the used execution platform to the maximum of its capabilities.

REFERENCES

1. A. Amer and C. Regazzoni. Introduction to the special issue on video object processing for surveillance applications. *Real-Time Imaging*, 11(3):167–171, 2005.
2. G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing: An International Journal*, 34:344–371, 1986.
3. L. F. Costa and R. M. C. Jr. *Shape Analysis and Classification.* CRC Press, 2001.
4. L. F. Costa, E. T. M. Manoel, F. Faucereau, J. van Pelt, and G. Ramakers. A shape analysis framework for neuromorphometry. *Network: Computation in Neural Systems*, 13(3):283–310, 2002.
5. O. Cuisenaire and B. Macq. Fast euclidean transformation by propagation using multiple neighborhoods. *Computer Vision and Image Understanding*, 76(2):163–172, 1999.
6. A. R. Dick and M. J. Brooks. Issues in automated visual surveillance. In *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications (DICTA 2003)*, pages 195–204, Sydney, Australia, December 2003.
7. S. F. F. Gibson. Calculating the distance map for binary sampled data. Technical Report TR99-26, Mitsubishi Electric Research Laboratories, 1999.

8. W. Guan and S. Ma. A list-processing approach to compute Voronoi diagrams and the Euclidean distance transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):757–761, 1998.
9. A. Hampapur, L. M. Brown, J. Connell, M. Lu, H. Merkl, S. Pankanti, A. W. Senior, C.-F. Shu, and Y.-L. Tian. Multi-scale tracking for smart video surveillance. *IEEE Transactions on Signal Processing*, 22(2):38–51, 2005.
10. R. Kimmel, N. Kiryati, and A. M. Bruckstein. Multivalued distance maps for motion planning on surfaces with moving obstacles. *IEEE Transactions on Robotics and Automation*, 14(3):427–436, 1998.
11. R. Kimmel, D. Shaked, N. Kiryati, and A. M. Bruckstein. Skeletonization via distance maps and level sets. *Computer Vision and Image Understanding*, 62(3):382–391, 1995.
12. Y. Lu, T. Jiang, and Y. Zang. Region growing method for the analysis of functional MRI data. *NeuroImage*, 20(1):455–465, 2003.
13. C. R. Maurer, Jr., R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.
14. F. Meyer. Topographic distance and watershed lines. *Signal Processing*, 38:113–125, 1994.
15. A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM*, 13(4):471–494, 1966.
16. A. Rosenfeld and J. L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.
17. T. E. Schouten, H. C. Kuppens, and E. L. van den Broek. Timed Fast Exact Euclidean Distance (tFEED) maps. *Proceedings of SPIE (Real Time Imaging IX)*, 5671:52–63, 2005.
18. T. E. Schouten, H. C. Kuppens, and E. L. van den Broek. Three dimensional fast exact euclidean distance (3D-FEED) maps. *Proceedings of SPIE (Vision Geometry XIV)*, 6066:60660F, 2006.
19. T. E. Schouten, H. C. Kuppens, and E. L. van den Broek. Video surveillance using distance maps. *Proceedings of SPIE (Real-Time Image Processing)*, 6063:54–65, 2006.
20. T. E. Schouten and E. L. van den Broek. Fast Exact Euclidean Distance (FEED) Transformation. In J. Kittler, M. Petrou, and M. Nixon, editors, *Proceedings of the 17th IEEE International Conference on Pattern Recognition (ICPR 2004)*, volume 3, pages 594–597, Cambridge, United Kingdom, 2004.
21. F. Y. Shih and J. J. Liu. Size-invariant four-scan euclidean distance transformation. *Pattern Recognition*, 31(11):1761–1766, 1998.
22. F. Y. Shih and Y.-T. Wu. Fast Euclidean distance transformation in two scans using a 3×3 neighborhood. *Computer Vision and Image Understanding*, 93(2):195–205, 2004.
23. E. L. van den Broek, T. E. Schouten, and P. Kisters. Modeling human color categorization. *Pattern Recognition Letters*, 10.1016/j.patrec.2007.09.006, 2007.
24. E. L. van den Broek, T. E. Schouten, P. M. F. Kisters, and H. C. Kuppens. Weighted Distance Mapping (WDM). In N. Canagarajah, A. Chalmers, F. Deravi, S. Gibson, P. Hobson, M. Mirmehdi, and S. Marshall, editors, *Proceedings of the IEE International Conference on Visual Information Engineering (VIE2005)*, pages 157–164, Glasgow, United Kingdom, 2005. Wrightsons - Earls Barton, Northants, Great Britain.
25. A. Zelinsky. A mobile robot navigation exploration algorithm. *IEEE Transactions of Robotics and Automation*, 8(6):707–717, 1992.